3rd ML-INFN Hackathon: *Advanced Level*

BAYESIAN HYPERPARAMETER OPTIMIZATION

and the *Hopaas* toolkit

Matteo Barbetti (INFN-Firenze, University of Firenze)

Lucio Anderlini (INFN-Firenze)

OUTLINE

1. **What is Bayesian optimization?**
   - *Graphical representation of Bayesian optimization*
2. **Bayesian techniques for Machine Learning**
   - *Strategies for hyperparameter optimization*
3. **Optuna**
   - *Framework for automatic optimization*
4. **Hyperparameter optimization as a service**
   - *The Hopaas toolkit*

1. WHAT IS BAYESIAN OPTIMIZATION?

Graphical representation of Bayesian optimization

Naive example: the gas mining problem

The goal is to mine for natural gas in an unknown land. For simplicity, we assume that the gas is distributed about a line. We want to find the location along this line with the maximum gas content while only drilling a few times since drilling is **expensive**. In addition, we assume some unknown tectonic phenomenon such that mining multiple times at the same point could result in a different amount of gas extracted.

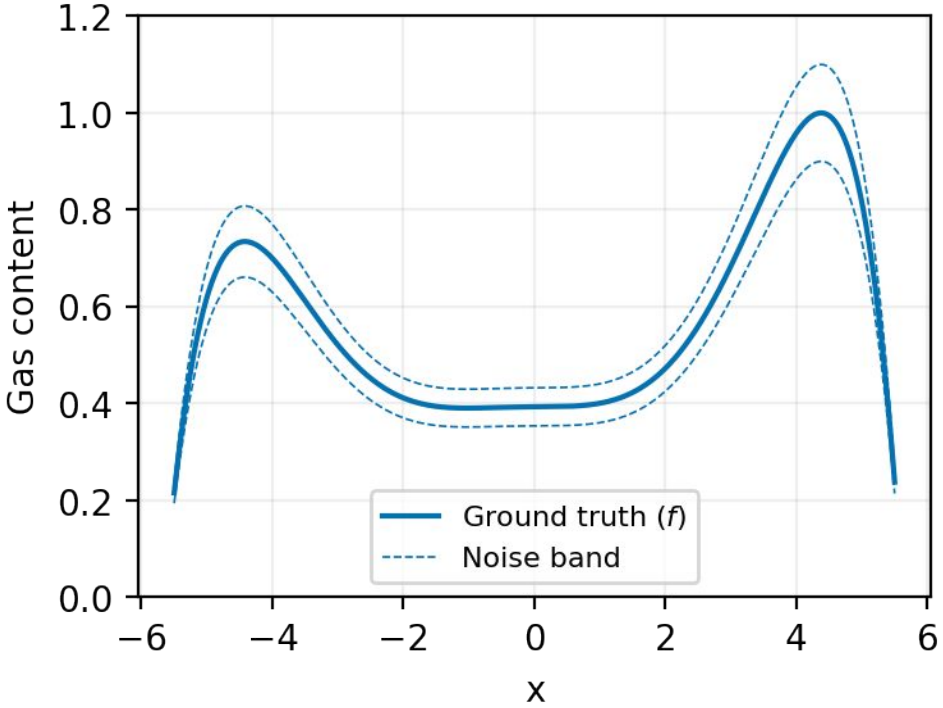
[*] Example freely adapted from the Distill post by A. Agnihotri and N. Batra.

A more general formulation of the problem

Let f(x) be the gas distribution (**objective function**), then this optimization problem is challenging because of:

- f(x) is unknown, often highly non-linear and probably non-convex;
- no gradient information comes from evaluations;
- evaluating f(x) is expensive;
- once evaluated, f(x) is known with **uncertainties**.

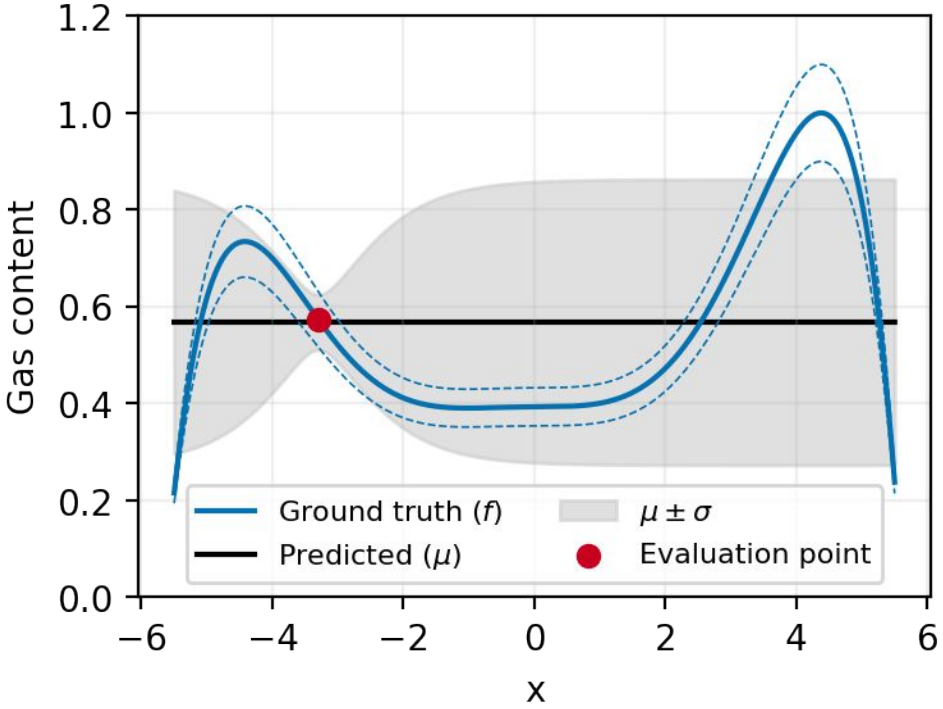
Representing the objective function



How to face the gas mining problem

To face this problem, we can approximate the objective function with a more tractable probabilistic model (called ***surrogate model***). The surrogate model can be initialized with our prior belief of f(x). As we evaluate points (drilling), we get more data for our surrogate to learn from, updating it according to **Bayes' rule** (posterior belief). In addition, we can build a surrogate model that takes into account the intrinsic uncertainties of the objective model.

Representing the surrogate model



How to choose new evaluation points

Since evaluating f(x) is expensive, we should choose the evaluation points smartly. We make this decision with something called ***acquisition function***. The acquisition function depends on our present knowledge of the objective function and suggests new points to evaluate in an exploration and exploitation trade-off:

- **Exploration** – evaluation of uncertain regions;
- **Exploitation** – evaluation of regions we already know have higher gas content.

Acquisition functions

In general, **acquisition functions** have these properties:

- they are heuristics for choosing the evaluation points;
- they are a function of the surrogate posterior;
- they combine exploration and exploitation;
- they are inexpensive to evaluate.

Sequential Model-Based Optimization

Defining an iterative procedure to update the surrogate model and select new evaluation points (according to an acquisition function) implements a Bayesian optimization strategy. This iterative procedure is typically named ***Sequential Model-Based Optimization*** (SMBO).

Example of surrogate models:

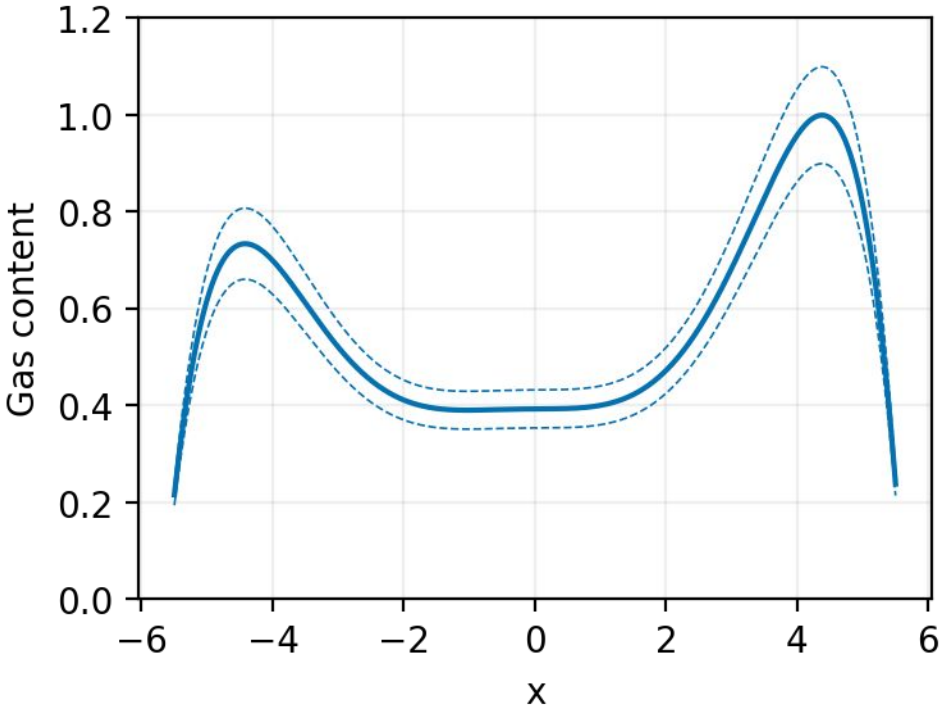
- Gaussian Processes
- Random Forest Regressions
- Tree-structured Parzen Estimators

Example of acquisition functions:

- Probability of improvement
- Expected improvement
- Upper confidence bounds

Representing SMBO

[i = 0]

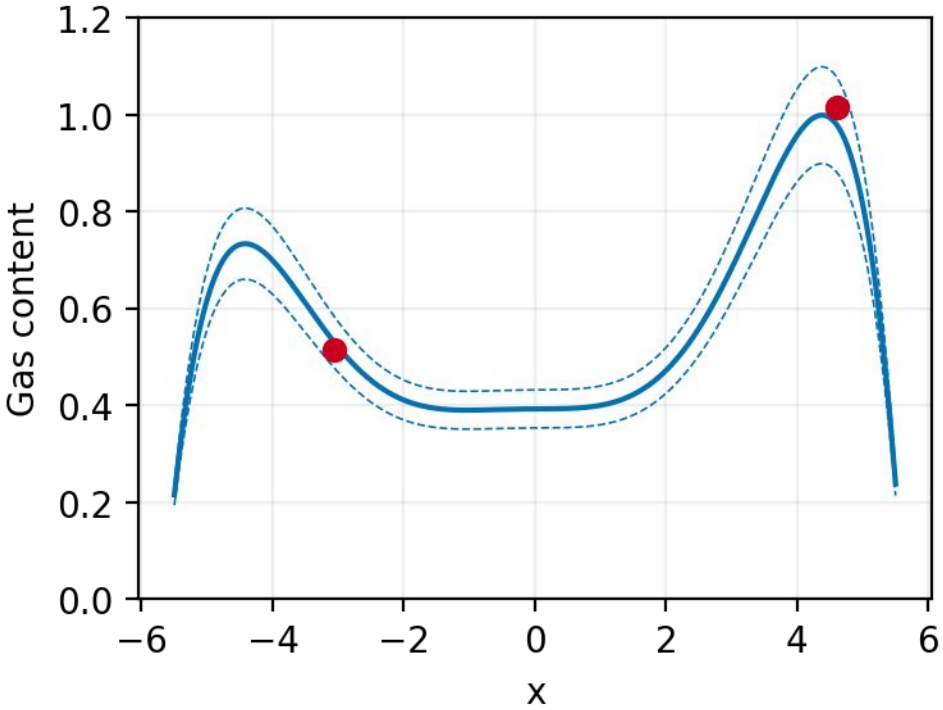


Objective function f(x):

- unknown
- non-linear
- non-convex
- expensive evaluation
- with uncertainties

Representing SMBO

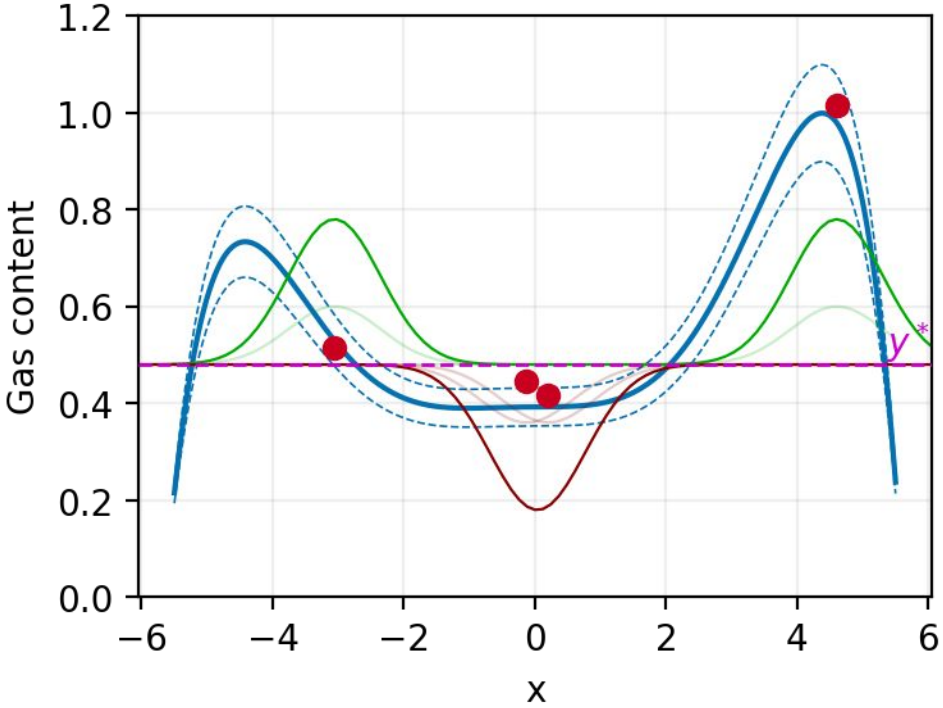
[i = 2]



Evaluation points sampled from a uniform distribution

Representing SMBO

[i = 4]



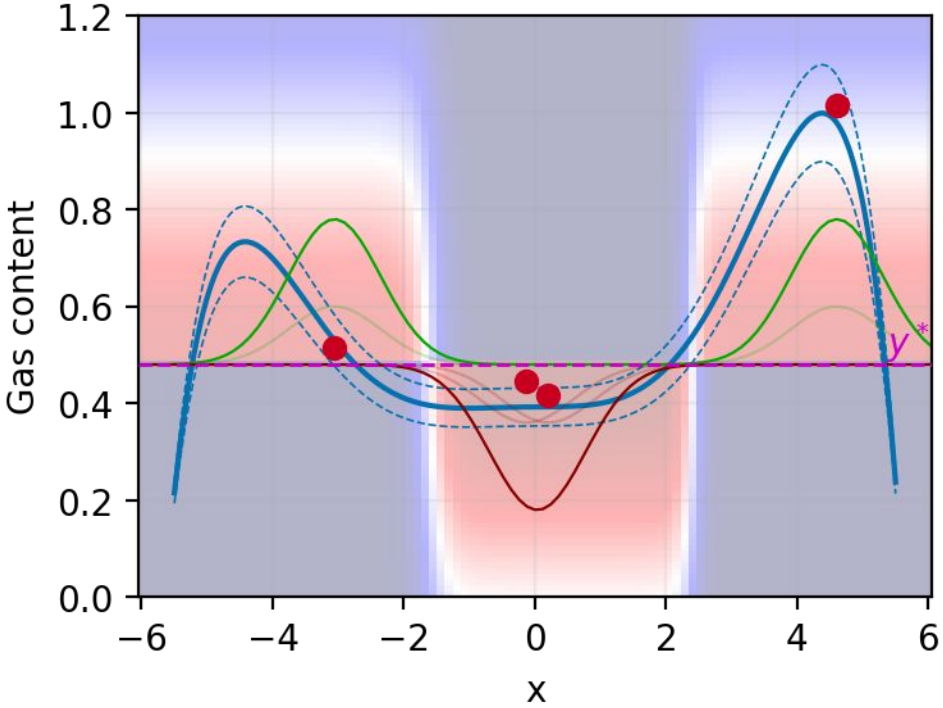
Evaluation points sampled from a uniform distribution

- y* = *reasonable* value for the objective function (median)
- TPE-based **surrogate model**
- likelihood defined as

$$p(x|y) = \begin{cases} \ell(x) & \text{if } y < y^* \\ g(x) & \text{if } y \geq y^* \end{cases}$$

Representing SMBO

[i = 4]



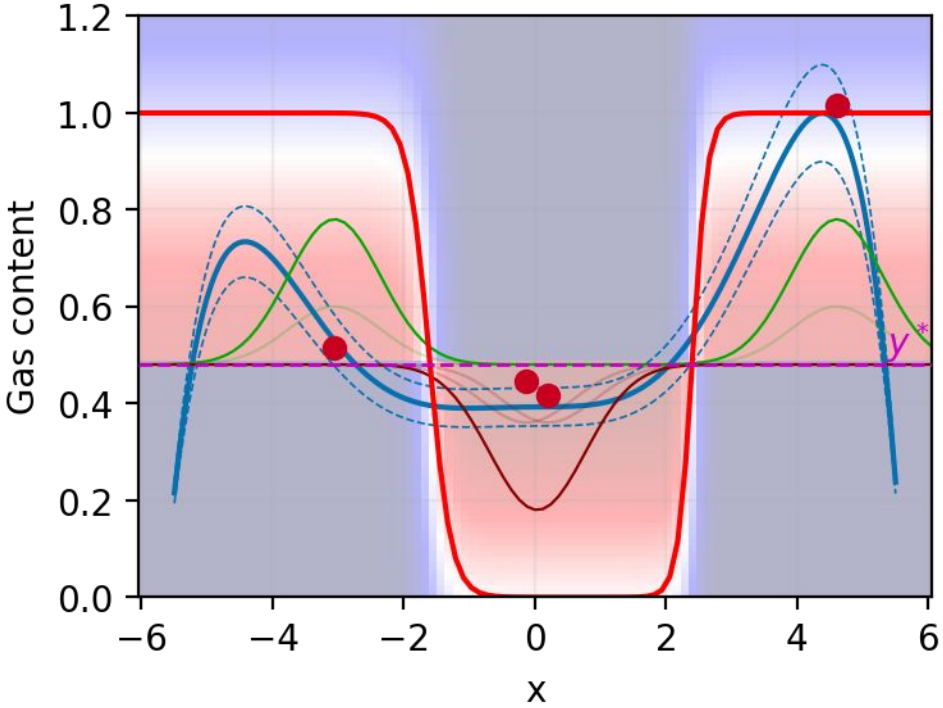
Evaluation points sampled from a uniform distribution

- y* = *reasonable* value for the objective function (median)
- TPE-based surrogate model represented as 2D histo
- surrogate model derived from the **Bayes' rule**

$$p(y|x) = \frac{p(x|y)\,p(y)}{p(x)}$$

Representing SMBO

[i = 4]



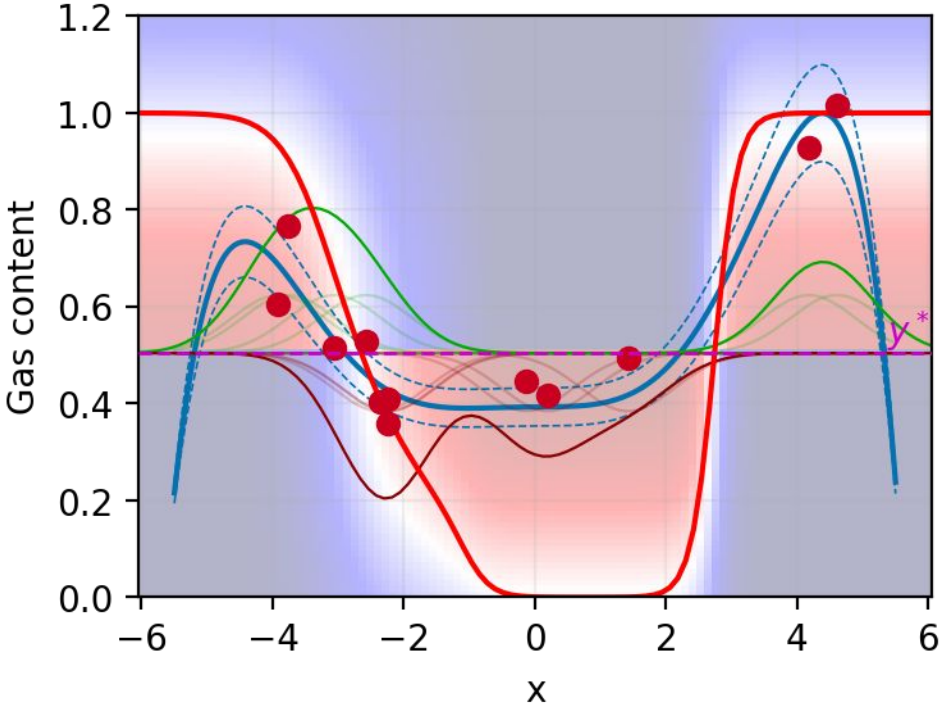
Evaluation points sampled from a uniform distribution

- y* = *reasonable* value for the objective function (median)
- EI-based **acquisition function**

$$\mathrm{EI}_{y^*}(x) = \int_{-\inf}^{y^*} (y^* - y)\frac{p(x|y)p(y)}{p(x)}dy = \ldots$$
$$\propto \left(\gamma + \frac{g(x)}{\ell(x)}(1-\gamma)\right)^{-1}$$

Representing SMBO

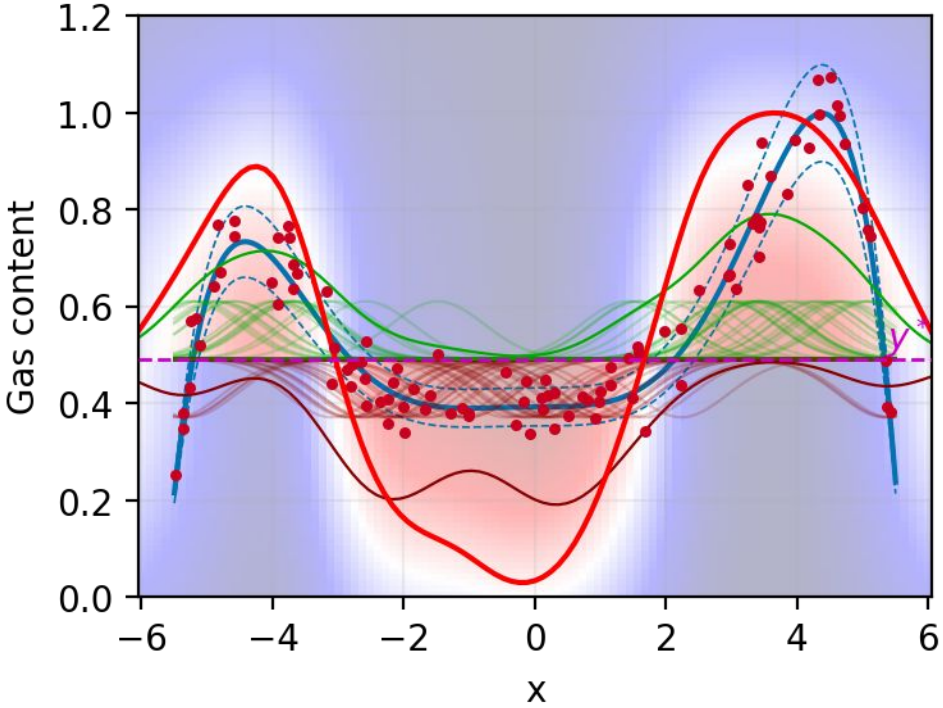
[i = 12]



Evaluation points sampled according to the acquisition function

Representing SMBO

[i >> 12]

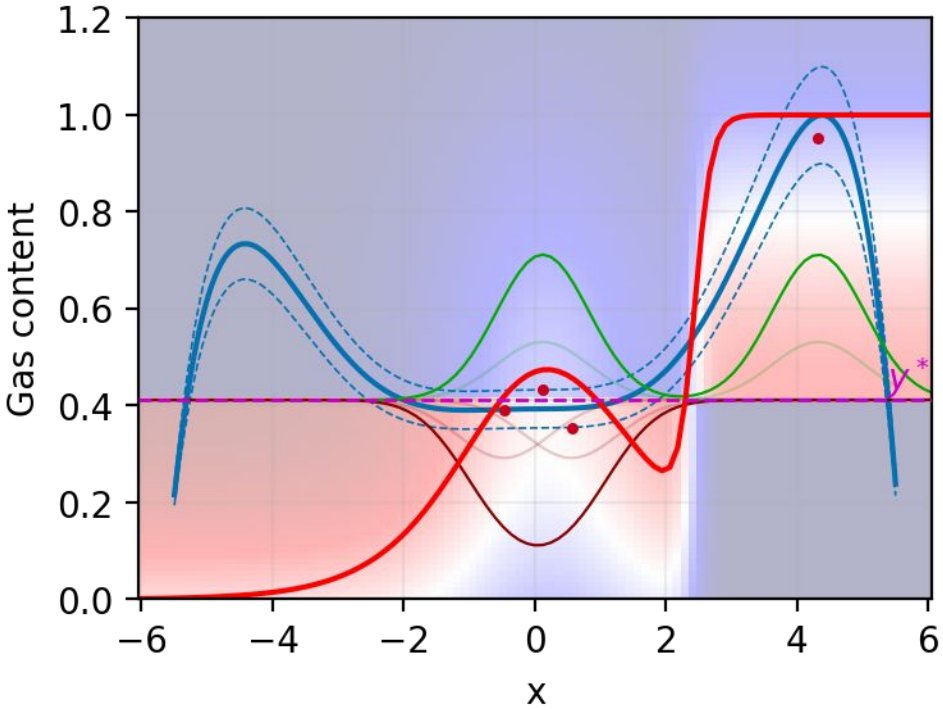


Evaluation points sampled according to the acquisition function

- several evaluations
- surrogate good approximator
- max{EI} = optimum region

Representing SMBO

[i = 4]



- Sampling too early from the acquisition function benefits exploitation against exploration
- Moving y* helps in **fixing** exploitation and exploration trade-off

2. BAYESIAN TECHNIQUES FOR MACHINE LEARNING

Strategies for hyperparameter optimization

Parameters VS Hyperparameters

Let X and Y be respectively the input and output datasets such that y=f(x) with f unknown. Using Machine Learning (ML), we want to find the best parameterization for f, namely

$$\hat{f}(x;\theta,\varphi) \doteq \hat{f}_\theta(x;\varphi)$$

where θ are the **parameters** of our ML model (automatically set during the training process), and φ are the **hyperparameters** that define such training procedure.

Hyperparameter optimization

To push the performance of our ML model we can define some score function S (i.e. accuracy, AUC, MSE, BCE, KS-test, ...) over a set of *controllable parameters* (namely, the hyperparameters) that we typically want to optimize (min/max):

$$\varphi^* = \arg\max_{\varphi \text{ in } \Phi} \mathcal{S}\left(y, \hat{f}_\theta(x;\varphi)\right)$$

This optimization problem is named ***hyperparameter optimization***.

Machine Learning (black-box) model

Machine Learning models are typically referred to as ***black-box* functions** whose optimization is a non-trivial problem, since

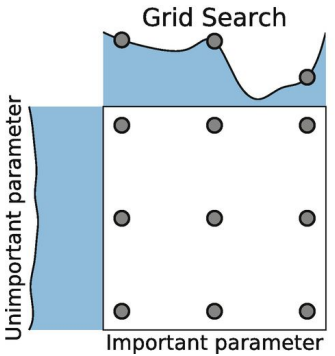
- models implement non-linear and non-convex functions;
- optimization is typically a (very) high-dimensional problem;
- testing a set of hyperparameters is an expensive task;
- evaluating twice same set of hyperparameters ends up with different results.

Grid Search and Random Search

Standard methods to search for good hyperparameter candidates are:

- **Manual**;
- **Grid Search** – exhaustive search over pre-specified range;
- **Random Search** – randomized hyperparameter search.

None of these methods exploit the whole history of tests!



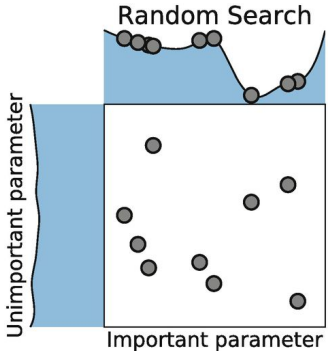


Figure stolen from the E. Lee's blog post

Bayesian hyperparameter optimization

Techniques like TPE suggest new sets of hyperparameters based on the present knowledge of the objective function. Contrary to Grid and Random searches, Bayesian optimization exploits all the tests performed through the **prior/posterior evolution**.

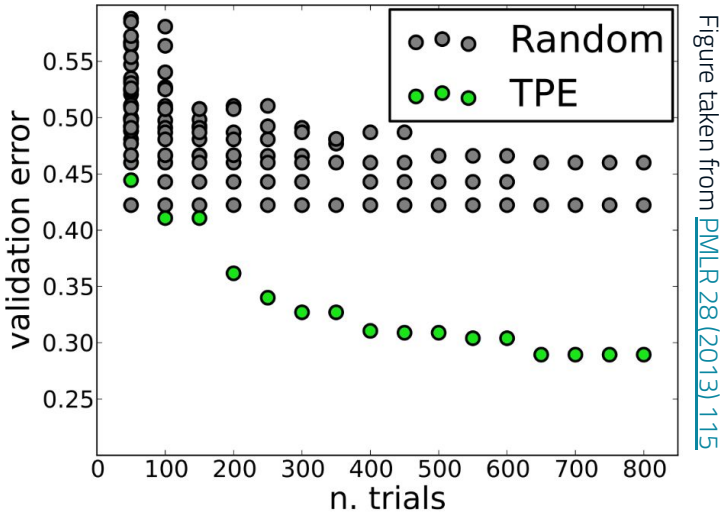


Figure taken from PMLR 28 (2013) 115

Bayesian optimization in Python

Bayesian optimization can be easily implemented in Python using, for instance, one of the following tools:

- Optuna – Framework to automate hyperparameter search;
- Weights & Biases – Framework to handle the whole ML model life cycle (model results storing, hyperparameter optimization, models and datasets versioning, web-based dashboards for developing and reporting);
- Ray Tune – Library for hyperparameter optimization designed to ease the multi-GPU and/or multi-node scale-up;
- BoTorch – Library for Bayesian optimization research built on top of PyTorch;
- scikit-optimize – Library for SMBO built on top of NumPy, SciPy, and Scikit-Learn.

3.
OPTUNA

Framework for automatic optimization

The Optuna framework

Optuna is an open-source framework designed for automatic hyperparameter optimization.

- Optuna is entirely written in Python and has few dependencies
  - The hyperparameter search space is defined taking into account the **Python syntax**, including conditionals and loops
- Optuna implements efficient optimization algorithms
  - The user has access to state-of-the-art algorithms for **sampling** set of hyperparameters and **pruning** efficiently unpromising trials
- Optuna enables **framework-agnostic** optimization campaigns
  - The user is free to use the preferred Machine Learning framework (i.e. Scikit-Learn, TensorFlow, PyTorch, JAX, ...)

[*] More details at https://optuna.org.

Main components: study and trials

For instance, let's optimize a simple quadratic function: $(x-2)^2$. Our goal is to find the value of `x` that minimizes the output of the `objective` function. During the optimization, Optuna repeatedly calls and evaluates the objective function with different values of `x`.

- A **`Trial`** object corresponds to a single execution of the objective function;
- A **`Study`** object represents an optimization session, which is a set of trials;
- In Optuna, a **parameter** is a variable whose value is to be optimized, such as `x`.

```
import optuna

def objective(trial):
  x = trial.suggest_float("x", -10, 10)    # defines an Optuna parameter
  return (x - 2) ** 2

study = optuna.create_study()              # builds an Optuna Study instance
study.optimize(objective, n_trials=100)    # calls 100 Optuna Trial instances

study.best_params    # e.g. {'x': 2.002108042}
```

Let's code!

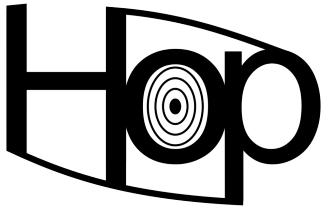
4. HYPERPARAMETER OPTIMIZATION AS A SERVICE

The *Hopaas* toolkit

The Hopaas toolkit

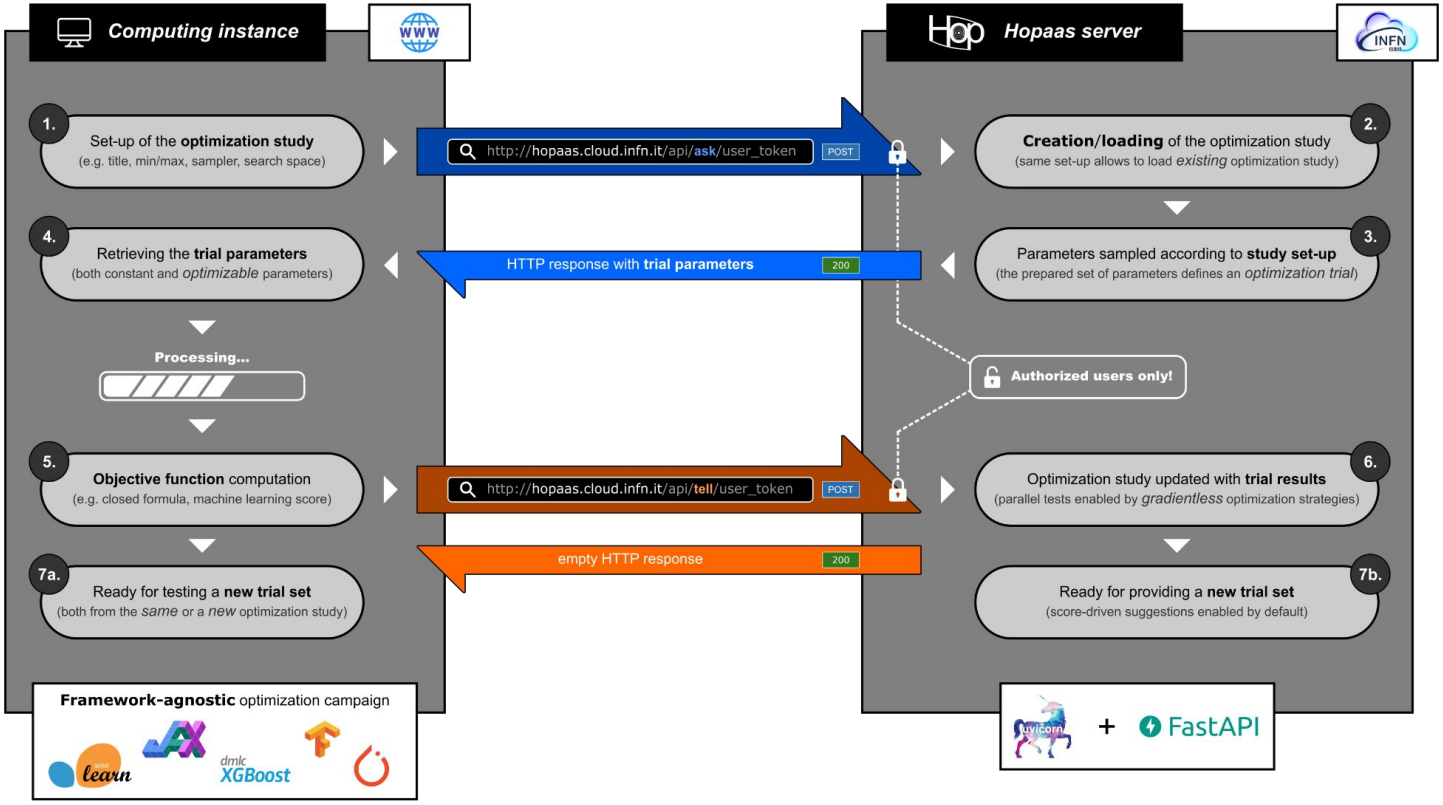
Hopaas (*Hyperparameter OPtimization As A Service*) is a service hosted by INFN Cloud that allows orchestrating optimization studies across multiple computing instances via **HTTP requests**. Hopaas provides a set of REST APIs and a web interface to enable **user authentication** and monitor the status of the user studies.

- **Back-end** based on FastAPI;
- Underlying databases powered by PostgreSQL;
- **Bayesian optimization** powered by Optuna;
- Service provided by Uvicorn and NGINX;
- Python **front-end** available (hopaas_client).

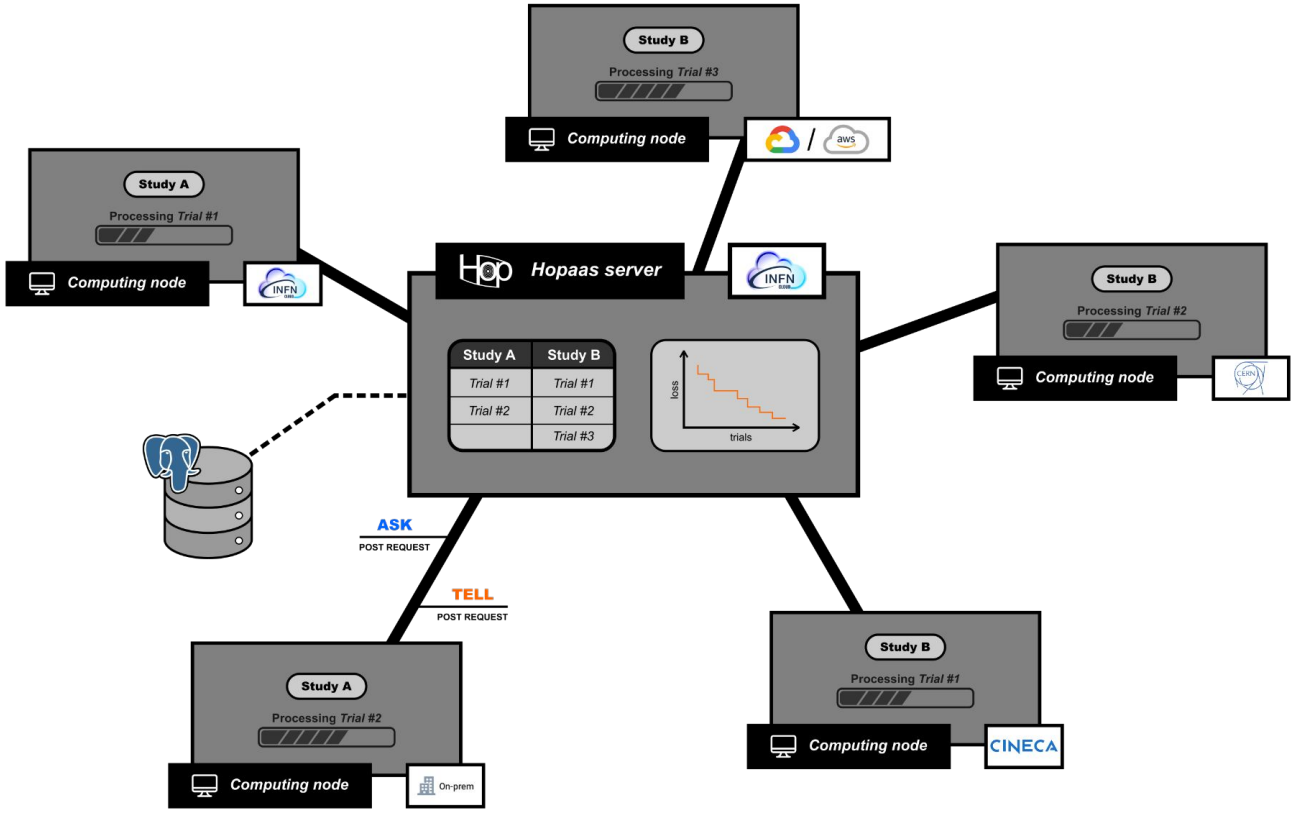


[*] More details at https://hopaas.cloud.infn.it.

Client-server optimization system



Multi-nodes optimization campaigns



Let's code!

CONCLUSIONS

- Bayesian strategies can be effectively used to find the global optimum point of a generic black-box function
  - *This problem is tackled probabilistically using the so-called **surrogate function***
- Hard hyperparameter optimizations are the perfect use-cases for **Bayesian optimization**
  - *The evaluation of the objective function is limited to regions with "high probability" to find optima*
- Several tools can be used to easily implement Bayesian optimization
  - *We have investigated the use of **Optuna** and **Hopaas** for single-node and multi-nodes optimization campaigns respectively*

THANKS!

Any questions?

You can find us at:

- Matteo.Barbetti@fi.infn.it
- Lucio.Anderlini@fi.infn.it

BACKUP

Tree-structured Parzen Estimator

Contrary to Gaussian Process (GP) that models the surrogate p(y|x) directly, ***Tree-structured Parzen Estimator*** (TPE) approximated the objective function in terms of p(x|y) and p(y). In particular, TPE defines p(x|y) as

$$p(x|y) = \begin{cases} \ell(x) & \text{if } y < y^* \\ g(x) & \text{if } y \geq y^* \end{cases}$$

Expected Improvement

Let's consider a minimization problem, then the ***Expected Improvement*** (EI) is defined by

$$\mathrm{EI}_{y^*}(x) = \int_{-\inf}^{y^*} (y^* - y) p(y|x) dy$$

where y* is a threshold value of the objective function, and p(y|x) the surrogate probability model. The aim is to **maximize EI** with respect to x. The value of y* controls the exploration and exploitation trade-off.